Sample Spaces Uniform on Neighborhoods

Leonard J. Schulman*
Department of Mathematics
Massachusetts Institute of Technology
Cambridge MA 02139

Abstract

Let a universe of m elements be given, along with a family of subsets of the universe (neighborhoods), each of size at most k. We describe methods for assigning the m elements to points in a small-dimensional vector space (over $GF(2)$), in such a way that the elements in each neighborhood are assigned to an independent set of vectors.

Such constructions lead, through a standard correspondence between linear and statistical independence, to the construction of small sample spaces which restrict to the uniform distribution in each neighborhood. (The sample space is a uniformly-weighted family of binary m-vectors).

The size of such a sample space will be a function of the number of neighborhoods; and for sparse families, will be substantially smaller than any space which restricts to the uniform distribution in all k-sets. Previous work on sample spaces with limited independence focused on providing independence or near-independence in every k-set of the universe.

We show how to construct the sample spaces efficiently both sequentially and in parallel. In case there are polynomially many (in m) neighborhoods, each of size $O(\log m)$, the parallel construction is in NC.

These spaces provide a new derandomization technique for algorithms; particularly, algorithms related to the Lovász local lemma. We also describe applications to the exhaustive testing of VLSI circuits, and to coding for burst errors on noisy channels.

1 Introduction

One of the most significant advances in the study of algorithms in recent years, has been the introduction of a new design technique, the method of derandomization. In this method, one first identifies an efficient randomized algorithm for the problem of concern. One views this randomized algorithm as sampling within the range of computations defined by its coin tosses — some substantial fraction of which are "good". In this light the goal of derandomization is to produce an efficient deterministic procedure to identify a good sample point (coin sequence).

Broadly, two approaches to derandomization have been studied. The first (Spencer [17], Raghavan [15]) supposes that, given any partial assignment to the coins (random variables), the conditional probability (or conditional expectation) of finding a good point, is easily computable to a sufficient accuracy. Then by sequentially setting the variables so as not to decrease the likelihood of success, one can construct a good assignment, in time that is proportional to the logarithm of the size of the sample space (and to the time required for each of the computations).

The second approach takes advantage of the fact that in many instances, the existence of a high proportion of good sample points does not hinge on total mutual independence of the random variables. Many algorithms have been shown to work as well,

*Supported in part by an ONR graduate fellowship, an MIT Applied Mathematics graduate fellowship, NSF Grant 8912586 CCR and Air Force Contract AFOSR 89-0271.

24th ANNUAL ACM STOC - 5/92/VICTORIA, B.C., CANADA
© 1992 ACM 0-89791-512-7/92/0004/0017...$1.50

or nearly as well, when only k-wise independence of the random variables is assumed. Thus the original sample space was replaced by a much smaller space, possessing k-wise independence; and then the algorithm was derandomized by searching that space exhaustively for a good sample point. This approach was introduced by Luby [11], who used a pairwise distribution to solve the Maximal Independent Set problem. An important feature of this approach is that it lends itself to highly parallel computation, since a separate processor can be assigned to check each sample point. Thus Luby placed several problems in NC. (For MIS this was already due to Karp and Widgerson [10]).

The size of a k-wise independent sample space on m variables is at least $m^{\lfloor k/2 \rfloor}$ ([7],[2]). Thus only when k is constant is a straightforward implementation of this approach possible in polynomial time (or, if parallelizable, in NC). (All random variables in our discussion will be binary). More sophisticated techniques were introduced by Berger and Rompel [6] and by Motwani, Naor and Naor [13] to handle certain classes of instances in which k is polylogarithmic in m.

Naor and Naor [14] made a key advance by observing that it is often sufficient for the applications to use a small-bias probability space; e.g. to require that any k variables of the space be *nearly* independent. They constructed polynomial size sample spaces which are nearly independent on any subset of $c \log m$ variables. By way of contrast note that a space which was precisely independent on every subset of size $c \log m$ would be of size $\Omega(m^{\log m})$. Their construction is in NC. Thus their method enabled them to both derandomize, and place problems in NC (e.g. finding heavy codewords in a linear code). Simpler constructions of sample spaces with similar properties were later provided by Alon, Goldreich, Hastad and Peralta [3].

We will take a different approach toward the derandomization of problems which require limited (e.g. up to $O(\log m)$-wise) independence. In various instances independence of the sample space is required only in a small designated family of k-sets of the variables. (We will refer to these designated k-sets as neighborhoods). We will construct sample spaces which are independent in each of an explicitly given list of neighborhoods. The size of the space will be 2^k times the greatest number of neighborhoods which any variable belongs to. A key example is the case of polynomially many neighborhoods each of size $O(\log m)$. In this case our sample space will be of polynomial size, and hence provide polynomial-time deterministic algorithms.

Our approach, when applicable, has two advantages over that of the Naor and Naor "near-independence" method:

(I) The space constructed is exactly independent within each neighborhood. Beyond its greater simplicity, this may be particularly useful in applications in which it is desired to sample repeatedly without compounding a deviation from the uniform distribution.

(II) The size of a nearly-independent space is proportional to a polynomial in $1/\epsilon$, where ϵ measures how near the distribution is to uniform. (ϵ = the max over assignments of the variables in any neighborhood, of $|2^{-k} - Pr(\text{the assignment})|$.) The size our sample space lacks entirely the $(1/\epsilon)^c$ multiplicative term needed, in the near-independence method, in order to ensure that the ϵ-bias is sufficiently small for the applications.

Let us examine a very simple example to illustrate the situation. Suppose m vertices are connected in a chain, and a binary variable is to be selected at each vertex. It is desired that at any pair of linked vertices, the assignments $(0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$ be equiprobable. There is a sample space of *constant* size — four — that satisfies the requirement. However any space which ignores the family of neighborhoods (the edges), and provides pairwise independence, will be of size $\Omega(m)$. A striking contrast persists even if one is willing to settle for pairwise near-independence (observe that $\Omega(\log m)$ strings are necessary just in order that all four assignments occur on all pairs of vertices).

Our sample spaces are efficiently constructible: for instance in the important case mentioned earlier, of polynomially many neighborhoods each of logarithmic size, our construction is in NC.

Our method relies upon the following well known correspondence between linear and statistical independence. Let $v_1, ..., v_m$ be vectors in a vector space V over $GF(2)$. Let a vector w be chosen randomly, uniformly in V; and let H be a subset of $\{1, ..., m\}$ such that $\{v_i\}_{i \in H}$ are an independent set of vectors. Then the binary random variables $\{\langle v_i, w \rangle\}_{i \in H}$ are mutually statistically independent. (Here $\langle \cdot, \cdot \rangle$ denotes dot product over $GF(2)$).

Thus, given a family of neighborhoods, our efforts will focus on the construction of a set of vectors, in as small a vector space as possible, such that the vectors in every neighborhood are independent. Put another way, a matrix M is to be constructed with few columns, so that every specified neighborhood of rows is independent.

In this perspective the matrix M acts as an operator on column vectors (the randomly chosen w). It has been observed in the literature that the dual perspective, as well, is very fruitful: a matrix M in which every k rows are independent, acts (on row vectors) as a parity-check matrix for a linear code with minimum distance k between codewords. (Thus u is a codeword if and only if $uM = 0$). Conversely any k rows of the parity-check matrix for such a code, must be independent. This viewpoint predates the sample-space application in the literature (see [12]); they were first brought together in in [7] and in [2].

The dual, coding-theoretic perspective extends naturally to arbitrary families of neighborhoods. We will describe a particularly important example, in which the matrices define codes that are very resilient to burst errors. The tendency of errors to come in runs, or bursts, afflicts almost every communication channel, simply because interferences are likely to have different, and possibly longer, durations than the transmission time for individual bits; and also because causally related interferences may cluster. The problem of coding for such errors has therefore received substantial attention. The model for burst errors which we will describe is more flexible than the standard guard-space model in the literature (see [9]). Our method lacks an efficient decoding procedure; but it may prove attractive if such an implementation can be devised.

In section 2 we establish the existence of matrices (and sample spaces) with the properties we seek. This existence proof is not needed for the explicit constructions presented later; but it illuminates, we hope, the parameters of the problem. In section 3 we describe a sequential construction, and in section 4 a parallel construction. Finally in section 5 we describe applications to the derandomization of the Beck/Alon algorithmic Lovász local lemma; to the testing of VLSI circuits; and to coding for burst errors on noisy channels.

2 An Existence Argument

Let $\mathcal{H}$ be a family of h subsets ("neighborhoods") of the collection of m binary variables $x_1, ...x_m$. (We also speak of the neighborhoods as simply being subsets of $\{1, ..., m\}$). Let k be the maximum size of a neighborhood in $\mathcal{H}$.

Suppose V is a vector space over $GF(2)$, and $v_1, ..., v_m$ are vectors in V with the property that for any neighborhood $H \in \mathcal{H}$, the vectors $\{v_i\}_{i \in H}$ are linearly independent. Then we define a sample space in the following way: pick a vector w from V uniformly, and set each variable x_i to the dot product $\langle v_i, w \rangle$. For any linearly independent $v_{i_1}, ..., v_{i_k}$, this induces the uniform distribution on assignments of $x_{i_1}, ..., x_{i_k}$.

The size of the sample space is the number of vectors in V, or $2^{\dim V}$. In the present section we will show on abstract grounds why a small-dimensional vector space V suffices for this construction. We defer an explicit construction (which is actually slightly more economical) to the next section.

Proposition 1 *Let $\mathcal{H}$ be a family of h neighborhoods in $\{1, ..., m\}$, each of size at most k. Let V be a vector space over $GF(2)$ of dimension $l = k + 1 + \lfloor \lg h \rfloor$. Then there exist vectors $v_1, ..., v_m \in V$ such that $\{v_i\}_{i \in H}$ are linearly independent for every neighborhood $H \in \mathcal{H}$.*

Proof: Select the vectors $\{v_i\}$ independently and uniformly in V. The probability that the vectors in a neighborhood H are independent is

$$(1 - 2^{-l})(1 - 2^{1-l})(1 - 2^{2-l}) \cdots (1 - 2^{(|H|-1)-l})$$

which is at least $1-2^{k-l}$. Applying a union bound, we find that the probability that there is any neighborhood in which the vectors are dependent, is at most $h2^{k-l} = h2^{-1-\lfloor \lg h \rfloor} < 1$. Thus there exists a satisfactory assignment of $v_1, ..., v_m$. □

Observe that for a particular neighborhood H, the event that the set of vectors in H are linearly dependent, is statistically independent of the collection of all such events concerning neighborhoods which do not intersect H. Therefore one can apply the Lovász local lemma ([17] §8) instead of a union bound in the above argument. If D is the maximum number of neighborhoods of $\mathcal{H}$ which intersect any individual neighborhood, then this argument shows that it suffices to take $\dim V = k + 1 + \lfloor \lg eD \rfloor$. (Which may or may not be an improvement over $k + 1 + \lfloor \lg h \rfloor$).

3 A Sequential Construction

As before let $\mathcal{H}$ be a family of h neighborhoods of the collection of m binary variables $x_1, ...x_m$, and let k be the maximum size of a neighborhood in $\mathcal{H}$. Further let d be the maximum number of neighborhoods containing any point.

In the following lemma we show how to construct a family of vectors which is linearly independent in every neighborhood of $\mathcal{H}$:

Theorem 1 *Given $\mathcal{H}$ as above, we can construct a set of vectors $v_1, ..., v_m$, in a vector space of dimension $\lceil k + \lg d \rceil$ over $GF(2)$, such that $v_1, ..., v_m$ are linearly independent in any neighborhood H of $\mathcal{H}$.*

The construction can be made in time linear in m, and polynomial in d and in 2^k.

Proof: Let V be a vector space of dimension $\lceil k + \lg d \rceil$ over $GF(2)$. We associate with each variable x_i a vector v_i in such a manner that the vectors within any neighborhood are linearly independent. We construct the vectors in sequence. At every stage we will ensure that the vectors within each neighborhood are linearly independent.

Suppose $v_1, ..., v_{j-1}$ have been defined already. In order to define v_j we examine each of the neighborhoods x_i belongs to. The vectors which have already been set within each such neighborhood H, span a subspace of V of dimension at most $k-1$. Any of the vectors in this subspace (there are at most 2^{k-1} of them) is a forbidden choice for v_j as it would introduce a dependence among the vectors in H. On the other hand the choice of any other vector v_j will leave the vectors in H linearly independent. We take a union bound: each neighborhood containing x_i rules out at most 2^{k-1} choices for v_j, and there are at most d such neighborhoods. Thus there are at most $d2^{k-1}$ forbidden choices for v_j. Since V contains at least $d2^k$ vectors, a satisfactory choice for v_j remains. □

This construction is more efficient (in terms of the dimension of V) than the existence argument of the previous section (whether using the union bound or the Lovász lemma).

Using the standard correspondence between linear and statistical independence, we now have:

Corollary 1 *There exists a sample space U of assignments to the variables $\{x_i\}$, of size at most $|U| = d2^{k+1}$, such that for any neighborhood H in $\mathcal{H}$, and for any setting of binary values to the variables in H, there are exactly $|U|/2^{|H|}$ points of U which restrict to that setting on the variables in H.*

The elements of U can be computed in time linear in m, and polynomial in d and 2^k. □

In particular for $k = O(\log m)$, U is constructed in time polynomial in the length of the input.

In any case, recall that U is being constructed for the purposes of a derandomized algorithm which will enumerate all its points in search of a "good" sample point. Hence the time to construct U, is always polynomially related to time complexity of the derandomized algorithm for which it is intended.

4 A Parallel Construction

The construction described in the previous section is inherently sequential. A more sophisticated technique is necessary in order to make the construction in parallel.

Our method will be to first construct a matrix M' over $GF(2)$ such that any k of the m rows of M' are linearly independent; thus, these rows would

serve well as the vectors $\{v_i\}$, except that they are too long (i.e. the associated sample space is too large). Then we will show how to construct a matrix R such that $M'R$ has much shorter rows (i.e. length $O(k + \log h)$), yet every set of rows with indices in a neighborhood H is still independent. Then $v_i = (M'R)_i$ will be our desired construction.

Let $k' = \max\{k, \log m\}$. Let W be a vector space of dimension k over $GF(2^{k'})$. For any $\alpha \in GF(2^{k'})$ let $w(\alpha) \in W$ be the vector $(\alpha, \alpha^2, ..., \alpha^k)$. (The moment curve construction, see [4][1].) Let $\alpha_1, ..., \alpha_m$ be distinct elements of $GF(2^{k'})$. Define the $m \times k$ matrix M over $GF(2^{k'})$ by letting its i'th row be $w(\alpha_i)$. It is a standard fact that any k rows of M are linearly independent.

Now fix an arbitrary irreducible polynomial f of degree k' over $GF(2)$, and represent each element of $GF(2^{k'})$ in terms of its coefficients modulo f. I.e. an element $a \in GF(2^{k'})$ is regarded as a polynomial $a_{k'-1}z^{k'-1} + ... + a_1 z + a_0$, and is represented by the vector $(a_{k'-1}, ..., a_0)$. (Coefficients $a_i \in GF(2)$).

Define the $m \times kk'$ matrix M' over $GF(2)$ by simply writing out each element of M in terms of its coefficient representation modulo f. Since no k rows of M are dependent over $GF(2^{k'})$, no k rows of M' are dependent over $GF(2)$. (I.e. no set of $j \leq k$ rows of M' sum to $\bar{0}$).

Our goal is to show how to construct in NC a $kk' \times O(k + \log h)$ matrix R over $GF(2)$ such that in the matrix $M'R$, every subset of rows whose indices are a neighborhood H of the family $\mathcal{H}$, is linearly independent. I.e. if all the indices of a nonempty subset of rows of $M'R$ lie within some neighborhood H, then those rows do not sum to $\bar{0}$. Let $\mathcal{H}'$ denote the collection of all such nonempty subsets. Note that $|\mathcal{H}'|$ is at most $h2^k$.

We will construct R column by column. At each stage we will ensure that the new column R^j "handles" a constant fraction of the remaining elements of $\mathcal{H}'$. For R^j to "handle" a set of rows $M_{i_1}, ..., M_{i_b}$ means that $\sum_{l=1}^{b} \langle M_{i_l}, R^j \rangle = 1$. Note that this means that the sum of the rows $(M'R)_{i_1}, ..., (M'R)_{i_b}$ has a 1 in the j'th entry, and thus these rows of $M'R$ do not sum to $\bar{0}$.

Since each new column of R will handle a constant fraction of the remaining elements of $\mathcal{H}'$, our construction will terminate after $O(\log |\mathcal{H}'|) \leq O(k + \log h)$ stages. This is important both for the time complexity of the construction, and because this is the length of the resulting vectors $v_i = (M'R)_i$.

[1] We describe the simplest option. A slightly more efficient construction, following Alon, Babai and Itai [2], is to use a binary BCH matrix.

Construction of R

For purposes of exposition let us simplify the situation by choosing a specific, interesting range of the parameters. Let us suppose that $k = k' = c_1 \log m = c_2 \log d$ for suitable constants c_1, c_2. Thus M' is an $m \times k^2$ matrix over $GF(2)$.

Observe that our condition that a column R^j "handle" a set of rows in $\mathcal{H}'$, namely $\sum_{l=1}^{b} \langle M_{i_l}, R^j \rangle = 1$, can be restated as $\langle \sum_{l=1}^{b} M_{i_l}, R_j \rangle = 1$. By assumption in this context, $\sum_{l=1}^{b} M_{i_l} \neq \bar{0}$. We will use a small family of vectors S in $(GF(2))^{k^2}$ such that:

- For any nonzero vector $v \in (GF(2))^{k^2}$ there is a constant fraction of vectors $s \in S$ such that $\langle v, s \rangle = 1$. (Here v is in the role of arbitrary nonzero $\sum_{l=1}^{b} M_{i_l}$).

Note that there is a trivial construction which satisfies this "constant fraction" property: the set of all vectors in $(GF(2))^{k^2}$. The fraction in this case is $1/2$. However this set is of size 2^{k^2}, which is far too big for our purposes. A satisfactory construction (related to "ϵ-biased" families, see also [18]) will be referred to below.

At any stage of the construction of R, we will have a collection of the remaining ("unhandled") elements of $\mathcal{H}'$, call it $\mathcal{H}'_{j-1}$ (where $\mathcal{H}'_0 = \mathcal{H}'$); and we will wish to find a vector R^j that handles a constant fraction of the elements of $\mathcal{H}'_{j-1}$. We will do this in parallel by assigning a separate machine to each pair $(s, \{i_1, ..., i_b\})$. (Here $s \in S$ and $\{i_1, ..., i_b\} \in \mathcal{H}'_{j-1}$). Each machine will check whether its vector s handles its element of $\mathcal{H}'_{j-1}$, i.e. whether $\langle \sum_{l=1}^{b} M_{i_l}, s \rangle = 1$. Due to the nature of the family S, a constant fraction of those machines assigned a particular member of $\mathcal{H}'_{j-1}$, give a postive answer to this question. Hence the same is true of a constant fraction of all the machines; and so there is a particular vector s which handles at least that fraction of the elements of $\mathcal{H}'_{j-1}$.

After the machines have run their checks (in time $O(k^2)$; or better, parallelized on $O(k^2)$ machines in time $O(\log k)$), such a vector s can be identified in parallel in time $O(\log(|S| \cdot |\mathcal{H}'|))$. We then let R^j be the vector s, let $\mathcal{H}'_j$ be those elements of $\mathcal{H}'_{j-1}$ which were not handled by s, and continue. As indicated previously, all vectors of $\mathcal{H}'$ will be handled within $O(\log |\mathcal{H}'|) \geq O(k + \log h) \geq O(\log m)$ iterations of this procedure, and that is the number of columns of the matrix R which we construct.

In order to demonstrate that this procedure lies in NC (for the stated range of k and d) we need a *small* family of vectors S in $(GF(2))^{k^2}$ satisfying the "constant fraction" property described above. Specifically, the size of S must be polynomial in 2^k. Furthermore, it must be constructible in NC.

Just such families — and in fact, even smaller families than we strictly require — were developed by Naor and Naor [14], and other constructions were later provided in [3]:

Lemma 1 [14, 3]: *There exists an NC-constructible family S of $O(k^4)$ vectors in $(GF(2))^{k^2}$, and a $\beta > 0$, such that for every nonzero $v \in (GF(2))^{k^2}$ at least $\beta|S|$ of the elements $s \in S$ satisfy $\langle v, s \rangle = 1$.* □

This concludes our parallel construction.

In particular, for the range of parameters we have concentrated on, we have shown:

Theorem 2 *Let polynomially many neighborhoods in $\{1, ..., m\}$ be given, each of size $O(\log m)$. Then we can construct in NC an $m \times O(\log m)$ matrix in which the rows indexed by any neighborhood, are linearly independent.* □

Corollary 2 *Let a set of m binary variables be given, along with any system of polynomially many neighborhoods in this set, each of size $O(\log m)$. Then we can construct in NC a polynomial size sample space, which restricts to the uniform distribution on the settings of the variables within each neighborhood.* □

The corresponding statement for arbitrary k and for arbitrarily many neighborhoods is:

Theorem 3 *Let h neighborhoods in $\{1, ..., m\}$ be given, each of size at most k. Then we can construct in time $O((k + \log h + \log m)^{O(1)})$, on $O(mh2^k(k + \log m)^{O(1)})$ processors, an $m \times O(k + \log h)$ matrix in which the rows indexed by any neighborhood, are linearly independent.* □

Corollary 3 *Let a set of m binary variables be given, along with any system of h neighborhoods in this set, each of size at most k. Then we can construct in time $O((k + \log h + \log m)^{O(1)})$ on $m(h2^k)^{O(1)}$ processors, a sample space of size $(h2^k)^{O(1)}$, which restricts to the uniform distribution on the settings of the variables within each neighborhood.* □

5 Applications

5.1 Derandomization of the Beck/Alon Algorithmic Lovász Local Lemma

Alon [1] provided a randomized, parallel version of Beck's algorithm [5] for identifying "good" assignments in many applications of the Lovász local lemma (e.g. hypergraph 2-coloring); and then showed that these problems are actually in NC^1 by derandomizing his procedure, using the near-independence method [14, 3].

Alternatively the randomized algorithm can be derandomized using our method. In parallel the previous method enjoys an advantage since, although our construction is in NC, we do not know if it can be produced in NC^1. In serial our approach has the advantage of lacking the $(1/\epsilon)^c$ complexity term of the near-independence approach. Both approaches provide polynomial time algorithms.

5.2 The Testing of VLSI Circuits

It is of considerable interest to be able to test whether a large integrated circuit is operating correctly. In principle the behavior of the circuit in response to all possible inputs, can be tested and compared with its desired behavior. However this is infeasible as the complexity of this procedure is exponential in the size of the circuit. Instead, substantial efforts (see [16] and references therein)

have focused on the assumption that the circuit is composed of many small components, each depending on relatively few inputs. In this case it suffices to exercise the circuit on a test set of inputs which exhausts the possible inputs to every component; while the behavior of each component is compared with its desired behavior. Such a test set can be much smaller than that required to exhaustively test the entire circuit. For instance, Naor and Naor describe a polynomial size test set for circuits with m inputs, subject only to the requirement that no component depend on more than $O(\log m)$ inputs.

We give an example of the application of our method. Assume that each component depends on only a constant number of inputs, and that every input affects up to a constant number of components. In this case we can construct a *constant* size test set for the circuit — without regard to the size of the circuit.

The inputs of the circuit correspond to the binary random variables of a sample space we construct; and the set of inputs to each component, is a neighborhood. Observe that a sample space which restricts to the uniform distribution within each of these neighborhoods will, in particular, range over all assignments within each neighborhood; hence such a sample space exhaustively tests the circuit. The sequential construction of section 3 provides us with such a sample space of constant size. This size is linear in the maximum number of components affected by any one input; and exponential (as it must be) in the number of inputs that enter a component.

5.3 Coding for Burst Errors on Noisy Channels

We briefly describe a standard model for burst errors on noisy channels. Our description is adapted from [9] §6.10. We suppose that a sequence of bits $x = x_1...x_m$ has been sent over a binary channel, and the sequence $y = y_1...y_m$ has been received at the other end. The error sequence is $z = z_1...z_m = x \oplus y$. The bursts in z are identified relative to a predefined positive integer, the *guard space* g. The bursts are defined as the intervals in $z_1, ..., z_m$ which contain no subinterval of g consecutive 0's, and which are minimal subject to this condition. Observe that there is a unique way to parse z into bursts relative to g.

A coding system is said to have burst error correcting capability b relative to guard space g, if it can correctly decode any message y that has been corrupted by a noise sequence z in which all bursts (relative to g) are of length at most b. An upper bound on the *rate* (in bits of information per binary transmission) of such a coding system is $(g - b)/(g + b)$. This can in fact be approached by algebraic coding techniques.

This model has the following weakness. Suppose the channel is afflicted with error bursts that are typically of length b_0. This b_0 cannot be used as the burst length b for the coding system, unless there is reason to believe that errors will not occur in close succession. Instead b must be chosen large enough so that, in view of assumptions regarding the channel (which are not explicit in the model), there is a vanishing probability that bursts would accumulate to length b.

We describe a model in which the burst length parameter corresponds to the burst length anticipated on the channel, and allowance is made for a flexible distribution of these bursts.

Say that an error sequence $z = z_1...z_m$ has b-weight w if all the 1's of z lie in the union of some w intervals, each of length b. (This is the norm of a distance function; for $b = 1$ it is the Hamming distance).

A code with blocks of length m will be said to be (δ, b) - burst error correcting, if it can correctly decode any message y that has been corrupted by a noise sequence z of b-weight δm. (Require for simplicity that δm be integer). For $b = 1$ this is simply a δm - error correcting code.

Theorem 4 *(a) A code of block-length m is (δ, b) - burst error correcting if and only if the difference between every pair of codewords is a sequence of b-weight greater than $2\delta m$.*

(b) A linear code is (δ, b) - burst error correcting if and only if every codeword is of b-weight greater than $2\delta m$. □

We describe an efficient (δ, b) - burst error correcting code on blocks of length m. Let a neighborhood family $\mathcal{H}$ consist of all subsets of $\{1, ..., m\}$

which can be written as the union of $2\delta m$ intervals, each of length b. Let M be a matrix with m rows, such that any set of rows with indices entirely within some member of $\mathcal{H}$, is linearly independent. Now we define a linear code with block length m, by specifying that M is its parity check matrix. The codewords are the vectors of the left nullspace of M; and any matrix whose rows are a basis for that nullspace, can serve as the encoding matrix. Observe that any codeword specifies a dependence among the rows of M, and therefore must have b-weight greater than $2\delta m$. Hence by theorem 4 this code is (δ, b) - burst error correcting.

We examine the efficiency of this coding system. (We will assume $\delta b \leq 1/4$.) From section 3 we know how to construct M with width $\lceil 2\delta mb + \lg(b\binom{m}{2\delta m-1})\rceil$; which we upper bound (excepting the rounding) by $m(2\delta b + H_2(2\delta))$. ($H_2$ denotes the entropy function in base 2.)

This is the number of check bits in every block of length m. Thus, the rate of this code is $1 - 2\delta b - H_2(2\delta)$.

This coding method must be compared with existing methods in two ways. First, we compare with the guard space method for coding for burst errors. Of course the true contrast between these methods lies in the modeling: in the guard space method the burst length reflects a worst case estimate on the length of an interrruption between clear transmission periods; in the present method the burst length reflects instead an estimate of a typical interruption.

Second, we compare our codes with codes which are at least as good at correcting errors, but which ignore the burst - error structure of the channel. A code with minimum distance $2\delta mb$ can achieve rate at least $1 - H_2(2\delta b)$, but no more than $1 - H_2(\delta b)$; the latter is the more interesting comparison as it is the rate at which almost every error sequence of weight δbm can be corrected; and thus, generally, of the most practical interest. In the accompanying figure we have plotted the rates $1-2\delta b-H_2(2\delta)$ and $1-H_2(\delta b)$ for a channel on which we wish to be able to correct up to δm bursts of length $b = 50$. The difference in the rates is due to the excessive error-correcting capability available in the code which ignores the burst structure of the errors.

Discussion

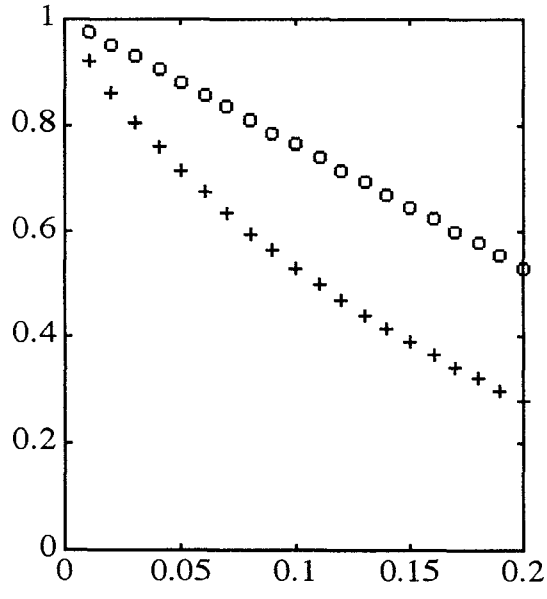
We do not know whether a parallel construction as in section 4 can be produced in NC^1 (for the case of polynomially many neighborhoods each of size $O(\log m)$) .

It would be of great interest to describe a coding system, for the burst error model described above, which possesses an efficient decoding algorithm.

Finally, for some very interesting, combinatorially motivated work regarding matrices in which specified subsets of the rows are linearly independent, see [8] and references therein.

Acknowledgements

I thank Michael Ben-Or, Peter Elias, Wayne Goddard, Mauricio Karchmer, Dan Kleitman and Mike Sipser for helpful discussions.



Abscissa is δb.
+ = rate ignoring burst structure.
o = rate for burst length b=50.

References

[1] N. Alon. A parallel algorithmic version of the local lemma. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pages 586–593, 1991.

[2] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7:567–583, 1986.

[3] N. Alon, O. Goldreich, J. Hastad, and R. Peralta. Simple constructions of almost k-wise independent random variables. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 544–553, 1990.

[4] László Babai and Péter Frankl. Linear algebra methods in combinatorics. Preliminary Version, 1988.

[5] J. Beck. An algorithmic approach to the Lovász local lemma I. *To appear in Random Structures and Algorithms.*

[6] B. Berger and J. Rompel. Simulating $(\log^c n)$-wise independence in NC. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 2–7, 1989.

[7] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich, and R. Smolenski. The bit extraction problem or t-resilient functions. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 396–407, 1985.

[8] Z. Füredi, J. R. Griggs, and D. J. Kleitman. Representations of families of triples over GF(2). *IMA Preprint Series # 422*, 1988.

[9] R. G. Gallager. *Information Theory and Reliable Communication.* Wiley, 1968.

[10] R. M. Karp and A. Wigderson. A fast parallel algorithm for the maximal independent set problem. *JACM*, 32(4):762–773, Oct. 1985.

[11] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, Nov. 1986.

[12] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* North-Holland, 1977.

[13] R. Motwani, J. Naor, and M. Naor. The probabilistic method yields deterministic parallel algorithms. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 8–13, 1989.

[14] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 213–223, 1990.

[15] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *JCSS*, 37(2):130–143, Oct. 1988.

[16] G. Seroussi and N. H. Bshouty. Vector sets for exhaustive testing of logic circuits. *IEEE Transactions on Information Theory*, 34(3):513–522, 1988.

[17] Joel Spencer. *Ten Lectures on the Probabilistic Method.* SIAM, 1987.

[18] U. V. Vazirani. *Randomness, Adversaries and Computation.* PhD Thesis, EECS, U.C. Berkeley, 1986.